2. RFID Tutorial: Introduction to RFID use and data handling

Gabrielle Davidson - School of Biological Sciences, University of East Anglia

2024-08-12

1. Get set up

Open R Studio

Check R packages installed

```
#load the following packages
library(tidyverse) #some of the below packages are included in tidyverse
library(rmarkdown)
library(dplyr)
library(tidyr)
library(lubridate)
library(ggplot2)
library(stringi) #may not need this
library(magrittr) #may not need this
```

Download resources necessary for the workshop

Download the resources from **my github page (https://github.com/DrGLDavidson/RFID-workshop)**

2.1 Introduction to RFID for animal behaviour

Considerations when designing RFID devices:

- Positioning of the RFID antenna (e.g. perch, see The Open Feeder (https://doi.org/10.1111/2041-210X.13931) for 3D printer file). At UEA we use “Selective Feeders” designed by Nature Counters (Dominic Goodwin), who has troubleshooted the positioning of the perch.
- Power – car batteries, AA batteries, solar panel. Temperature is going to affect power draw. At UEA we use “Selective Feeders” designed by Nature Counters (Dominic Goodwin), which takes 5 AA batteries.
- How many times RFID sends a signal and times when it’s dormant. Secondary detection method with low power that initiates RFID. Nature Counters use a strain gauge on the perch to initiate the RFID. Nestboxes use photodiodes to initiate the RFID.
- Water ingress.
- Squirrels, cattle, people etc.

Saving data in the field:

Depending on your RFID output, you’ll likely have information on what RFID device the data comes from. If it doesn’t, or even if it does, devices are mobile, so you must keep good records of where that device was located at the time of data collection and name your file/folder accordingly.

- Have a naming system for your RFID equipment and data storage device (e.g. SD card) and write it on outside of these devices, and name them electronically within them.
- Depending on your RFID equipment, you may require a laptop on site to download data from the device, or to deploy/initiate the program/change the program settings/update the time and date. Ideal scenario is a display where this can be edited manually (e.g. time/date), and data storage that can be swapped out (e.g. SD card) so you don't need to carry a laptop.
- I encourage you to keep a written notebook in the field to write time/date every time an RFID device is deployed, the name of the SD card notebook in the field when it was deployed and when it was removed
- If you come to a device to see that it has failed, make a log of this. Determining whether a device failed from the RFID visits is not a very reliable method to know if it failed or not because it could be because birds weren't visiting for example. Ideally a firmware program that has additional data files that have performance-based output (e.g. RFID signal, battery power) can allow you to know for sure when failures occurred and why. We suggest a readme.txt file describing the issues, any troubleshooting steps etc. This should be saved in a folder associated with that feeder dataset.
- Come ready with a backup device to swap out with failed devices that cannot be troubleshooted in the field and need to be brought back to the 'workshop'.
- When saving output, use a standardised file name system that contains information about the location/treatment. I suggest all files start with the date and should be in the format year/month/day so they appear chronologically in your folders. Alternatively you may want them sorted by location/nestbox, so start the file name with that info (e.g. "NB01_220427)

Data management

- The best practice is to work off a single master database always. If you have multiple copies of different formats you're prone to errors accumulating after each database is created, and forgetting which database was filtered in which way. Deviations from this are justifiable if you want your database to be organised according to site/nest box, in which case you are working from subsets of master data. Just be clear to label these as the master data files and always pull from these when running analyses or generating larger datasets.
- Option 1: Wait for all data to come in and create a single database file when all files are available
- Option 2: Create a master database as each file comes in throughout the field season. This option is idea because you are likely going to need to analyse data as the experiment is happening, otherwise you are doing an experiment "blind" and you may detect issues that can be corrected at the time.

In both cases, and especially in Option 2, there is a risk that you accidentally duplicate datasets if you merge it with file containing multiple datasets from different dates. *It is crucial that you keep a record of the code and the files imported so you can error check*. As we work through examples you will see just how important this is and why we will be using R Markdown and using the knit function each time data is imported and master datasets are generated.

Another option is to create an R function. This means you specify arguments (e.g. file names, dataframes) at the top of the function, rather than editing multiple lines of code to accommodate the appropriate files/dataframes. The latter can lead to errors if you don't update one or more of the dataframe or file names accordingly.

2.2 Importing RFID data and generating master

database

2.2.1 working directory and files

Imagine you have collected data from three RFID feeders deployed in the field. When you save the data from the SD card, you save it according to the date you collected it (YYMMDD), and the feeder ID (in this case F01, F02, F03)

```
require("knitr")
opts_knit$set(root.dir = "F:/RWorkspace/GitHub/RFID-workshop/data/feederData")

#set working directory so R knows where to find and save data
setwd("F:/RWorkspace/GitHub/RFID-workshop/data/feederData")

#specify the working directly as the object "path"

path<-"F:/RWorkspace/GitHub/RFID-workshop/data/feederData"

#use the function list.dirs to see all the folders in your directory
list.dirs(path, full.names = TRUE, recursive = TRUE)

#now use the function list.files to see what files are in that folder and all subfolders (recur
sive=TRUE).
list.files(path, pattern=NULL, all.files=FALSE,
    full.names=FALSE, recursive = TRUE)

#now use the function to look into one of the subfolders

path2<-"F:/RWorkspace/GitHub/RFID-workshop/data/feederData/F01_220206"
list.files(path2, pattern=NULL, all.files=FALSE,
    full.names=FALSE, recursive = TRUE)
```

what you are seeing are a number of files that start with a alphanumberic code, followed by two letters. “C1935” refers to the feeder’s hardware ID code. This can never change. The last two letters refer to the output file type. For example, FD refers to “feeder door” and has information about whether the servo door opened (o) or stayed closed (c) upon detection of a bird. For our purposes, we will focus on the RT files only.

2.2.2 Viewing files and preparing for compiling them into a single dataframe

```
#the files are text files so you need to use a function that calls .txt. files

setwd(path2) #since we already created path2 as folder F01_220206, lets start with that.
getwd() #double check you are now in the appropriate working directory
```

```
## [1] "F:/RWorkspace/GitHub/RFID-workshop/data/feederData/F01_220206"
```

```
# Call the RT file into your environment and call it df1 (dataframe 1)
df1<-read.delim("C1935RT.txt", header=TRUE)

names(df1) #see the column headers
```

```
##  [1] "F"         "Date"      "Hmsec"     "ID"        "Event"     "Channel"
##  [7] "Dur"       "Clks"      "Freq"      "Edges"     "Reps"      "Type"
## [13] "TagID_hex" "Mfr"       "TagID"
```

```
head(df1) # see the first few rows and headers
```

```
##    F                Date Hmsec     ID Event Channel    Dur  Clks   Freq Edges
## 1 D  2022-02-04 09:16:45   196 c1935     82       0 114855 61559 126720     1
## 2 D  2022-02-04 09:17:11   896 c1935     83       0 106855     0 125184     0
## 3 D  2022-02-04 09:17:11   648 c1935     83       0 114855    27 126208   156
## 4 D  2022-02-04 09:17:12   650 c1935     83       0 106855    37 126208   843
## 5 D  2022-02-04 09:18:30   270 c1935     84       0      2 62210 126208     1
## 6 D  2022-02-04 09:20:09   667 c1935     85       0 119946 17899 126464     1
##   Reps Type  TagID_hex Mfr TagID
## 1    1    0             NA    NA
## 2    0    0             NA    NA
## 3    1    1 0300024FEF  NA    NA
## 4    1    0             NA    NA
## 5    1    0             NA    NA
## 6    1    0             NA    NA
```

what you are seeing are time stamped events every time the strain gauge perch is triggered. If there is an associated RFID tag read, it appears under the column header “TagID_hex”. You can see the ID is c1935 which matches our folder name. Dur Clks Freq and Edges refer to some RFID parameters, which may be useful if there are any issues with detecting tags (there are lots of events without tags). the most important headers for downstream analyses are Date, Hmsec, ID and TagID_hex. But we will keep all headers. you should notice that although we have the ID of the feeder, that does not provide us with any information about where the feeder was located in our experimental design which is fundamental to analyses. If you are to compile a dataframe that contains data from multiple different feeders, you must know which data comes from which feeder.

2.2.3 Adding a column to indicate which feeder your data belongs to

```
#first look at your df1 in your global environment. It should say 337 obs. (observations) of 15
variables.

df1<-cbind(df1, feeder='F01') #add a column named feeder and have all the values as F01

#now df1 should say 16 variables. But do a sanity check:

head(df1)
```

```
##    F                Date Hmsec     ID Event Channel    Dur  Clks   Freq Edges
## 1 D  2022-02-04 09:16:45   196 c1935     82       0 114855 61559 126720     1
## 2 D  2022-02-04 09:17:11   896 c1935     83       0 106855     0 125184     0
## 3 D  2022-02-04 09:17:11   648 c1935     83       0 114855    27 126208   156
## 4 D  2022-02-04 09:17:12   650 c1935     83       0 106855    37 126208   843
## 5 D  2022-02-04 09:18:30   270 c1935     84       0      2 62210 126208     1
## 6 D  2022-02-04 09:20:09   667 c1935     85       0 119946 17899 126464     1
##   Reps Type  TagID_hex Mfr TagID feeder
## 1    1    0             NA    NA    F01
## 2    0    0             NA    NA    F01
## 3    1    1 0300024FEF  NA    NA    F01
## 4    1    0             NA    NA    F01
## 5    1    0             NA    NA    F01
## 6    1    0             NA    NA    F01
```

```
#we can repeat this for more feeders. We need to specify a new path and working directory, lets do that for feeder 2, with the same date.

path3<-"F:/RWorkspace/GitHub/RFID-workshop/data/feederData/F02_220206"
setwd(path3)

#double check you are in the correct directory
getwd()
```

```
## [1] "F:/RWorkspace/GitHub/RFID-workshop/data/feederData/F02_220206"
```

```
#if you want to check the file names:

list.files(path3, pattern=NULL, all.files=FALSE,
    full.names=FALSE, recursive = TRUE)
```

```
##  [1] "C1931AD.TXT" "C1931DG.TXT" "C1931FD.TXT" "C1931LA.TXT" "C1931LB.TXT"
##  [6] "C1931LC.TXT" "C1931LH.TXT" "C1931LI.TXT" "C1931LV.TXT" "C1931PA.TXT"
## [11] "C1931PE.TXT" "C1931RT.TXT" "C1931SY.TXT" "C1931W1.TXT" "logger.ini"
```

```
df2<-read.delim("C1931RT.txt", header=TRUE)

names(df2) #see the column headers
```

```
##  [1] "F"         "Date"      "Hmsec"     "ID"        "Event"     "Channel"
##  [7] "Dur"       "Clks"      "Freq"      "Edges"     "Reps"      "Type"
## [13] "TagID_hex" "Mfr"       "TagID"
```

```
head(df2) # see the first few rows and headers
```

```
##    F                Date Hmsec     ID Event Channel    Dur   Clks   Freq Edges
## 1 D  2022-01-21 08:29:28   977 c1931  15519       0 122438 161617 124416  3482
## 2 D  2022-01-25 09:28:29  1011 c1931  15520       0  44301  30396 137984     1
## 3 T  2022-02-04 09:12:44   232 c1931  15521      NA     NA     NA     NA    NA
## 4  S                        NA           NA      NA     NA     NA     NA    NA
## 5 D  2022-01-25 09:28:44   246 c1931  15521       0    908 998690 124672  5130
## 6 D  2022-02-04 09:16:04   843 c1931  15522       0 130917  18204 164608     1
##   Reps Type TagID_hex Mfr TagID
## 1   20    0           NA    NA
## 2    1    0           NA    NA
## 3   NA   NA           NA    NA
## 4   NA   NA           NA    NA
## 5   84    0           NA    NA
## 6    1    0           NA    NA
```

```
#this file has the same headers as df1. Lets add the feeder column and values.

df2<-cbind(df2, feeder='F02') #add a column named feeder and have all the values as F02
names(df2) #see the column headers
```

```
##  [1] "F"         "Date"      "Hmsec"     "ID"        "Event"     "Channel"
##  [7] "Dur"       "Clks"      "Freq"      "Edges"     "Reps"      "Type"
## [13] "TagID_hex" "Mfr"       "TagID"     "feeder"
```

```
head(df2) # see the first few rows and headers
```

```
##    F                Date Hmsec     ID Event Channel    Dur   Clks   Freq Edges
## 1 D  2022-01-21 08:29:28   977 c1931  15519       0 122438 161617 124416  3482
## 2 D  2022-01-25 09:28:29  1011 c1931  15520       0  44301  30396 137984     1
## 3 T  2022-02-04 09:12:44   232 c1931  15521      NA     NA     NA     NA    NA
## 4  S                        NA           NA      NA     NA     NA     NA    NA
## 5 D  2022-01-25 09:28:44   246 c1931  15521       0    908 998690 124672  5130
## 6 D  2022-02-04 09:16:04   843 c1931  15522       0 130917  18204 164608     1
##   Reps Type TagID_hex Mfr TagID feeder
## 1   20    0           NA    NA    F02
## 2    1    0           NA    NA    F02
## 3   NA   NA           NA    NA    F02
## 4   NA   NA           NA    NA    F02
## 5   84    0           NA    NA    F02
## 6    1    0           NA    NA    F02
```

```
#we can repeat this for the 3rd feeder too. We can also skip having to look in the directory by doing the following:

path4<-"F:/RWorkspace/GitHub/RFID-workshop/data/feederData/F03_220206"
setwd(path4)

RTfile<-list.files(path4, pattern = "RT")
df3<-lapply(RTfile, read.delim) #this will appear as a list in your environment
df3<-as.data.frame(df3) #turn it into a dataframe
head(df3)
```

```
##    F                Date Hmsec     ID Event Channel    Dur  Clks   Freq Edges
## 1 D  2022-02-04 09:16:34   121 c1937   2738       0 109071 16446 126464     1
## 2 D  2022-02-04 09:16:54   271 c1937   2739       0 106788     0 125952     0
## 3 D  2022-02-04 09:16:54    73 c1937   2739       0 113812    27 125952   156
## 4 D  2022-02-04 09:18:20   171 c1937   2740       0 109091 10831 126464     1
## 5 D  2022-02-04 09:21:12   445 c1937   2741       0 109071     0      0     0
## 6 D  2022-02-04 09:23:36   421 c1937   2742       0 109071 33838 126208     1
##   Reps Type  TagID_hex Mfr TagID
## 1    1    0             NA    NA
## 2    0    0             NA    NA
## 3    1    1 0300024FEF  NA    NA
## 4    1    0             NA    NA
## 5    1    0             NA    NA
## 6    1    0             NA    NA
```

```
df3<-cbind(df3, feeder='F03') #add a column named feeder and have all the values as F03
```

2.2.4 bind rows from different data frames into a single dataframes

```
#Because you have now indicated which feeder each line of data comes from, you can safely combi
ne all dataframes into one. They all have the same number of headers, and the headers are spell
ed the same, so this can be done without any additional changes.

d220206<-bind_rows(df1,df2,df3)  #I have named this according to the date of the folder these f
iles came from.

#you should now see d220206 in your environment and there should be 1526 observations

377+780+369  #the combined number of observations from df1, df2 and df3 =1526, which is expecte
d.
```

```
## [1] 1526
```

```
#you should now be able to do this for the other folder dated 220206. Because you may use a sim
ilar naming system, or copy and paste the code above and edit the code, it is good practice to
remove the dataframes that are no longer needed.

rm(df1,df2,df3) #this should remove them from your environment.
```

2.3 EXERCISE

There are three more files in the folder that come from different feeders that need to be added to the master database.

Make sure to

- add the feeder name column
- save a new Master database with **THE DATE OF THE LAST SET OF FILES YYMMDD**

```
#if you use the same naming system as before, you will then end up with a dataframe named d220209.

#bind rows for your dataframes from different dates

dfinal<-bind_rows(d220206,d220209)  #the order of the dfs as arguments will determine the order in which they appear in the dataframe. You can change this, but you could here choose the most contemporary date to come second.
```

2 .4.1 Saving your work and returning to it later

```
#by default, R will save to your current working directory. So lets change it to a higher directory than we are currently in. Otherwise it may be hard to find which subfolder to save into. You could also create a folder specifically for saving into.

setwd("F:/RWorkspace/GitHub/RFID-workshop/data/feederData")

write.table(dfinal, file = "Masterdf_220209.txt",sep="\t",row.names=FALSE)  #we are saving the dataframe called dfinal as a txt file into our working directory

#you should be able to view this using the following code:
list.dirs(path = "F:/RWorkspace/GitHub/RFID-workshop/data/feederData", full.names = TRUE, recursive = FALSE) #notice that recursive has been changed to F as we only want to see what is in that particular folder, not subfolders.

#Save your R script, and if using github, update the current branch, generate a pull request and merge to your main branch.
#when you close R, it may ask you if you want to save the workspace image. This means you can open that saved file with the data and values from your global environment as you left off. You can select no if that's not relevant. But if you were in the middle of something and needed to stop, that is a useful option.

#If you would like to clear your global environment entirely.
rm(list = ls())
```

Use R Markdown to keep track of your data importing and saving

Perhaps in the previous exercise you saved time by cutting and pasting repetitive code and changing the df and feeder number values?. What if you forgot to change the feeder number, or one of the df names? If you are going to do that, make sure you copy and paste, rather than write over code (i.e. replacing df1 with df2 but not keeping the script where you were working with df1). It is important to keep a record of every code you run. But there's still the possibility that over the course of repetitive copy-pasted code, you lose track of what was done when. R markdown is an excellent tool for keeping track of what you have done and sharing your code with others. Later, as you become more familiar with R you will also be able to write your own functions that will also minimise the chance of code errors/typos.

But for now, lets try R Markdown:

- In R studio go to File>New File> R Markdown…
- Input a title and click ok.
- Delete all the default text except the title

- On the top right hand corner of the R markdown window is a green square with a plus sign and “c”. Click that arrow and chose r script
- You can now write your r script in this gray area
- Use the “#” below r script to leave yourself notes
- write outside of the r script for headings or any other detail you want.
- click the Knit button on the top left hand side of the R markdown window and it will produce a document and save it to your working directory.

Additional resources for independent discovery of concepts and code: - google - stack overflow - chatgpt - R Markdown Cheat Sheet (https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf)

2.5 EXERCISE

Using the code you produced earlier, create a short R Markdown file and knit it.

Answers to EXERCISE

```
#the files are text files so you need to use a function that calls .txt. files

path2<-"F:/RWorkspace/GitHub/RFID-workshop/data/feederData/F01_220209"
setwd(path2)
# Call the RT file into your environment and call it df1 (dataframe 1)

RTfile<-list.files(path2, pattern = "RT")
df1<-lapply(RTfile, read.delim) #this will appear as a list in your environment
df1<-as.data.frame(df1) #turn it into a dataframe
head(df1)
df1<-cbind(df1, feeder='F01')

path3<-"F:/RWorkspace/GitHub/RFID-workshop/data/feederData/F02_220209"
setwd(path3)
# Call the RT file into your environment and call it df2 (dataframe 2)

RTfile<-list.files(path3, pattern = "RT")
df2<-lapply(RTfile, read.delim) #this will appear as a list in your environment
df2<-as.data.frame(df2) #turn it into a dataframe
head(df2)
df2<-cbind(df2, feeder='F02')

path4<-"F:/RWorkspace/GitHub/RFID-workshop/data/feederData/F03_220209"
setwd(path4)
# Call the RT file into your environment and call it df2 (dataframe 3)

RTfile<-list.files(path4, pattern = "RT")
df3<-lapply(RTfile, read.delim) #this will appear as a list in your environment
df3<-as.data.frame(df3) #turn it into a dataframe
head(df3)
df3<-cbind(df3, feeder='F03')

d220209<-bind_rows(df1,df2,df3)
```

END OF 2. RFID Tutorial: Introduction to RFID use and data handling

```
sessionInfo()
```

```
## R version 4.2.3 (2023-03-15 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19045)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.utf8
## [2] LC_CTYPE=English_United Kingdom.utf8
## [3] LC_MONETARY=English_United Kingdom.utf8
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.utf8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.43      magrittr_2.0.3  stringi_1.7.12  rmarkdown_2.22
##  [5] lubridate_1.9.2 forcats_1.0.0   stringr_1.5.0   dplyr_1.1.2
##  [9] purrr_1.0.1     readr_2.1.4     tidyr_1.3.0     tibble_3.2.1
## [13] ggplot2_3.4.2   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] bslib_0.5.0      compiler_4.2.3   pillar_1.9.0     jquerylib_0.1.4
##  [5] tools_4.2.3      digest_0.6.31    timechange_0.2.0 jsonlite_1.8.4
##  [9] evaluate_0.21    lifecycle_1.0.3  gtable_0.3.3     pkgconfig_2.0.3
## [13] rlang_1.1.0      cli_3.6.1        rstudioapi_0.14  yaml_2.3.7
## [17] xfun_0.39        fastmap_1.1.1    withr_2.5.0      hms_1.1.3
## [21] generics_0.1.3   sass_0.4.6       vctrs_0.6.1      grid_4.2.3
## [25] tidyselect_1.2.0 glue_1.6.2       R6_2.5.1         fansi_1.0.4
## [29] tzdb_0.4.0       scales_1.2.1     htmltools_0.5.5  colorspace_2.1-0
## [33] utf8_1.2.3       munsell_0.5.0    cachem_1.0.8
```